

CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU

PPPPPPPP		AAAAAA		RRRRRRRR		SSSSSSSS		EEEEEEEEEE	11
PPPPPPPP		AAAAAA		RRRRRRRR		SSSSSSSS		EEEEEEEEEE	11
PP	PP	AA	AA	RR	RR	SS		EE	1111
PP	PP	AA	AA	RR	RR	SS		EE	1111
PP	PP	AA	AA	RR	RR	SS		EE	11
PP	PP	AA	AA	RR	RR	SS		EE	11
PPPPPPPP		AA	AA	RRRRRRRR		SSSSSS		EEEEEEEE	11
PPPPPPPP		AA	AA	RRRRRRRR		SSSSSS		EEEEEEEE	11
PP		AAAAAAAAAA		RR	RR		SS	EE	11
PP		AAAAAAAAAA		RR	RR		SS	EE	11
PP		AA	AA	RR	RR		SS	EE	11
PP		AA	AA	RR	RR		SS	EE	11
PP		AA	AA	RR	RR		SS	EE	11
PP		AA	AA	RR	RR	SSSSSSSS		EEEEEEEEEE	111111
PP		AA	AA	RR	RR	SSSSSSSS		EEEEEEEEEE	111111

```

LL               IIIIII             SSSSSSSS
LL               IIIIII             SSSSSSSS
LL               II                SS
LL               II                SS
LL               II                SS
LL               II                SS
LL               II                SSSSSS
LL               II                SSSSSS
LL               II                SS
LL               II                SS
LL               II                SS
LL               II                SS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS

```



```

1 0001 0 MODULE parse1 (IDENT='V04-000'
2 0002 0 ADDRESSING_MODE(EXTERNAL=GENERAL))
3 0003 1 = BEGIN
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
9 0009 1 * ALL RIGHTS RESERVED.
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
16 0016 1 * TRANSFERRED.
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
20 0020 1 * CORPORATION.
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1 ++
29 0029 1 Facility: Command Definition Utility, CLD Parser Module 1
30 0030 1
31 0031 1 Abstract: This module is one of a few modules that implements the
32 0032 1 parser for CLD files. This parser translates the CLD source
33 0033 1 language into an intermediate representation composed of
34 0034 1 nodes linked in a directed graph.
35 0035 1
36 0036 1 Environment: Standard CDU environment.
37 0037 1
38 0038 1 Author: Paul C. Anagnostopoulos
39 0039 1 Creation: 30 November 1982
40 0040 1
41 0041 1 Modifications:
42 0042 1
43 0043 1 V04-001 BLS0270 Benn Schreiber 9-FEB-1984
44 0044 1 Correct IMAGE statement when image name is not quoted.
45 0045 1
46 0046 1 --
47 0047 1
48 0048 1
49 0049 1 library 'sys$library:lib';
50 0050 1 require 'clitabdef';
51 0375 1 require 'cdureq';

```

53	0789	1	!	T A B L E   O F   C O N T E N T S
54	0790	1	!	-----
55	0791	1	!	-----
56	0792	1	forward routine	
57	0793	1	cdu\$cld: novalue,	
58	0794	1	cdu\$statement,	
59	0795	1	cdu\$define_verb,	
60	0796	1	cdu\$define_syntax,	
61	0797	1	cdu\$define_type,	
62	0798	1	cdu\$sv_s_clause;	
63	0799	1		
64	0800	1	!	E X T E R N A L   R E F E R E N C E S
65	0801	1	!	-----
66	0802	1	!	-----
67	0803	1	external routine	
68	0804	1	cdu\$bool_expr,	
69	0805	1	cdu\$check_for_children,	
70	0806	1	cdu\$cli_flag,	
71	0807	1	cdu\$create_node,	
72	0808	1	cdu\$get_next_token,	
73	0809	1	cdu\$lookup_child,	
74	0810	1	cdu\$param_clause,	
75	0811	1	cdu\$qual_clause,	
76	0812	1	cdu\$report_syntax_error,	
77	0813	1	cdu\$token_must_be,	
78	0814	1	cdu\$type_clause,	
79	0815	1	cli\$present;	
80	0816	1		
81	0817	1	external	
82	0818	1	cdu\$gl_token_class: long,	
83	0819	1	cdu\$gq_token: descriptor;	



PARSE1  
V04-000

1 7  
15-Sep-1984 23:46:44  
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[CCDU.SRC]PARSE1.B32;1 Page 3  
(3)

```
: 85
: 86
: 87
: 88
: 89
: 90
: 91
: 92
: 93

0820 1 !      G L O B A L   D A T A
0821 1 !      -----
0822 1 !
0823 1 ! The following items contains the address of the root node of the
0824 1 ! intermediate representation. This node must be available to all other
0825 1 ! modules.
0826 1
0827 1 global
0828 1      cdu$gl_root_node: ref node;
```

OVERALL STRUCTURE

```

: 95      0829 1 :
: 96      0830 1 :
: 97      0831 1 :
: 98      0832 1 : This is a recursive descent compiler. That means that the source language,
: 99      0833 1 : as contained in CLD files, is compiled by routines which recognize certain
100      0834 1 : subsets of the language and call other such routine to recognize the rest
101      0835 1 : of the language. Routines may be invoked recursively when the syntax
102      0836 1 : is recursive.
103      0837 1 :
104      0838 1 : The language is assumed to be LL(1), which means it is parsed from left
105      0839 1 : to right and each construct can be recognized by inspecting only its
106      0840 1 : first token, with no backtracking. It is not strictly LL(1), but the
107      0841 1 : exceptions aren't too bad.
108      0842 1 :
109      0843 1 : The complete syntax for CLD files is presented below. Terminals are
110      0844 1 : shown in upper case, while nonterminals are shown in lower case. There
111      0845 1 : is a parsing routine for each nonterminal. This routine is responsible
112      0846 1 : for pulling all tokens for its construct from the file, and creating
113      0847 1 : one or more nodes which are the intermediate representation of the construct.
114      0848 1 :
115      0849 1 : The lexical and syntactic portions of the CDU are based primarily on work
116      0850 1 : done by A. Davie and R. Morrison, described in their book 'Recursive Descent
117      0851 1 : Compiling'.
118      0852 1 :
119      0853 1 : The intermediate representation of the file is composed of a set of nodes
120      0854 1 : linked together as a directed graph. Each node represents a single semantic
121      0855 1 : entity, with any subordinate entities as its children. Children are
122      0856 1 : linked together in a sister chain, as opposed to having an array of
123      0857 1 : child pointers in the node. The right-hand side of the syntax description
124      0858 1 : shows how the syntax is mapped into nodes.

```



			S Y N T A X	F O R	C L D	F I L E S	
			-----	----	----	-----	
126	0859	1	NONTERMINAL	SYNTAX			INTERMEDIATE REPRESENTATION
127	0860	1					
128	0861	1					
129	0862	1					
130	0863	1					
131	0864	1	cld ::=	{[,] statement}*			node with children
132	0865	1					
133	0866	1	statement ::=	IDENT h-string ;			node with string
134	0867	1		MODULE symbol ;			node with symbol
135	0868	1		define-verb ;			
136	0869	1		define-syntax ;			
137	0870	1		define-type			
138	0871	1					
139	0872	1	define-verb ::=	DEFINE VERB symbol {[,] v-s-clause}*			node with symbol, children
140	0873	1					
141	0874	1	define-syntax ::=	DEFINE SYNTAX symbol {[,] v-s-clause}*			node with symbol, children
142	0875	1					
143	0876	1	define-type ::=	DEFINE TYPE symbol {[,] type-clause}*			node with symbol, children
144	0877	1					
145	0878	1	v-s-clause ::=	CLIFLAGS ( cli-flag {, cli-flag}* ) ;			node with children
146	0879	1		CLIROUTINE symbol ;			node with symbol
147	0880	1		DISALLOW bool-expr ;			node with child
148	0881	1		NODISALLWS ;			node
149	0882	1		IMAGE h-string ;			node with string
150	0883	1		OUTPUTS ( symbol {,symbol}* ) ;			node with children
151	0884	1		PARAMETER Pn {[,] param-clause}* ;			node with symbol, children
152	0885	1		NOPARAMETERS ;			node
153	0886	1		PREFIX symbol ;			node with symbol
154	0887	1		QUALIFIER symbol {[,] qual-clause}* ;			node with symbol, children
155	0888	1		NOQUALIFIERS ;			node
156	0889	1		ROUTINE symbol ;			node with symbol
157	0890	1		SYNONYM symbol			node with symbol
158	0891	1					
159	0892	1	param-clause ::=	PROMPT [=] {symbol ; string} ;			node with string
160	0893	1		common_clause			
161	0894	1					
162	0895	1	qual-clause ::=	BATCH ;			node
163	0896	1		NEGATABLE ;			node
164	0897	1		NONNEGATABLE ;			node
165	0898	1		PLACEMENT [=] {GLOBAL ; LOCAL ; POSITIONAL} ;			node with symbol
166	0899	1		common_clause			
167	0900	1					
168	0901	1	type-clause ::=	KEYWORD symbol {[,] keyword-clause}* ;			node with symbol, children
169	0902	1		PREFIX symbol			node with symbol
170	0903	1					
171	0904	1	keyword-clause ::=	NEGATABLE ;			node
172	0905	1		NONNEGATABLE ;			node
173	0906	1		common_clause			
174	0907	1					
175	0908	1	common-clause ::=	CLIFLAGS ( cli-flag {, cli-flag}* ) ;			node with children
176	0909	1		DEFAULT ;			node
177	0910	1		LABEL [=] symbol ;			node with symbol
178	0911	1		SYNTAX [=] symbol ;			node with symbol
179	0912	1		VALUE [( value_clause {, value_clause}* )]			node with children
180	0913	1					
181	0914	1	value-clause ::=	CONCATENATE ;			node
182	0915	1		NOCONCATENATE			node

```

: 183 0916 1 : DEFAULT [=] {h-string ; node with children
: 184 0917 1 : ( string {, string}* )} ;
: 185 0918 1 : IMPCAT ; node
: 186 0919 1 : LIST ; node
: 187 0920 1 : REQUIRED ; node
: 188 0921 1 : TYPE [=] {builtin-type ; symbol} node with code or symbol
: 189 0922 1 :
: 190 0923 1 : builtin-type ::= one of a set of symbols beginning with $
: 191 0924 1 :
: 192 0925 1 : cli-flag ::= nodes
: 193 0926 1 :
: 194 0927 1 : ABBREVIATE ;
: 195 0928 1 : FOREIGN ;
: 196 0929 1 : IMMEDIATE ;
: 197 0930 1 : MCRIGNORE ;
: 198 0931 1 : MCROPTDELIM ;
: 199 0932 1 : MCRPARSE ;
: 200 0933 1 : NOSTATUS
: 201 0934 1 : bool-expr ::= bool-term {OR bool-term}* node with children
: 202 0935 1 : bool-term ::= bool-factor {AND bool-factor}* node with children
: 203 0936 1 :
: 204 0937 1 : bool-factor ::= ( bool-expr ) ; bool-expr node
: 205 0938 1 : ANY2 ( path {, path}* ) ; node with children
: 206 0939 1 : NOT path ; node with child
: 207 0940 1 : path
: 208 0941 1 :
: 209 0942 1 : path ::= [< symbol >] symbol {, symbol}* node with children
: 210 0943 1 :
: 211 0944 1 : string ::= " anything-but-quote* "
: 212 0945 1 : symbol ::= { $ | 0-9 | A-Z | _ | a-z }+
: 213 0946 1 : h-string ::= string ;
: 214 0947 1 : anything-but-open-close-comma-equal-eol+
: 215 0948 1 :
: 216 0949 1 :
: 217 0950 1 : NOTES:
: 218 0951 1 : Equal signs are optional in cases where they were documented
: 219 0952 1 : in the V3.0 documentation.
: 220 0953 1 : There is no provision for recognizing a number as a separate token.

```



```

222 0954 1  !++
223 0955 1  ! Description: This parsing routine recognizes the distinguished nonterminal
224 0956 1  ! "cld", which represents the entire CLD file.
225 0957 1  !
226 0958 1  ! Parameters: None.
227 0959 1  !
228 0960 1  ! Returns: Nothing.
229 0961 1  !
230 0962 1  ! Notes:
231 0963 1  ! --
232 0964 1  !
233 0965 1  GLOBAL ROUTINE cdu$cld          : novalue
234 0966 2  = BEGIN
235 0967 2
236 0968 2  local
237 0969 2      statement: ref node,
238 0970 2      last_statement: ref node;
239 0971 2
240 0972 2
241 0973 2  ! Create a root node for the tree.
242 0974 2
243 0975 2  cdu$gl_root_node = cdu$create_node(node_k_root);
244 0976 2
245 0977 2  ! Loop once for each statement in the CLD file.
246 0978 2
247 0979 2  cdu$get_next_token();
248 0980 2  until token_is(tkn_k_eof) do (
249 0981 2
250 0982 2      ! Statements may be separated with commas.
251 0983 2
252 0984 2      skip_optional_token(tkn_k_comma);
253 0985 2
254 0986 2      ! We must have another statement. Parse it and link its node
255 0987 2      ! onto the end of the statement list.
256 0988 2
257 0989 2      statement = cdu$statement(cdu$gl_root_node);
258 0990 2      link_parent_to_child(cdu$gl_root_node, statement, last_statement);
259 0991 2  );
260 0992 2
261 0993 2  return;
262 0994 2
263 0995 1  END;

```

INFO#250 L1:0990  
Referenced LOCAL symbol LAST\_STATEMENT is probably not initialized

```

.TITLE PARSE1
.IDENT \V04-000\

.PSECT $GLOBALS,NOEXE,2

00000 CDU$GL_ROOT_NODE::
.BLRB 4

.EXTRN CDU$BOOL_EXPR, CDU$CHECK_FOR_CHILDREN
.EXTRN CDU$CLI_FLAG, CDU$CREATE_NODE
.EXTRN CDU$GET_NEXT_TOKEN

```

			003C 00000	.EXTRN	CDU\$LOOKUP_CHILD	
				.EXTRN	CDU\$PARAM_CLAUSE	
				.EXTRN	CDU\$QUAL_CLAUSE	
				.EXTRN	CDU\$REPORT_SYNTAX_ERROR	
				.EXTRN	CDU\$TOKEN_MUST_BE	
				.EXTRN	CDU\$TYPE_CLAUSE	
				.EXTRN	CLISPRESENT, CDU\$GL_TOKEN_CLASS	
				.EXTRN	CDU\$GQ_TOKEN	
				.PSECT	\$CODE\$,NOWRT,2	
				.ENTRY	CDU\$CLD, Save R2,R3,R4,R5	: 0965
				MOVAB	CDU\$GL_ROOT_NODE, R5	
				MOVAB	CDU\$GET_NEXT_TOKEN, R4	
				PUSHL	#1	: 0975
				CALLS	#1, CDU\$CREATE_NODE	
				MOVL	R0, CDU\$GL_ROOT_NODE	
				CALLS	#0, CDU\$GET_NEXT_TOKEN	: 0979
				MOVL	CDU\$GL_TOKEN_CLASS, R0	: 0980
				CMPL	R0, #4	
				BEQL	5\$	
				CMPL	R0, #5	: 0984
				BNEQ	2\$	
				CALLS	#0, CDU\$GET_NEXT_TOKEN	
				PUSHL	CDU\$GL_ROOT_NODE	: 0989
				CALLS	#1, CDU\$STATEMENT	
				MOVL	R0, STATEMENT	
				MOVL	CDU\$GL_ROOT_NODE, R0	: 0990
				TSTL	8(R0)	
				BNEQ	3\$	
				MOVL	STATEMENT, 8(R0)	
				BRB	4\$	
				MOVL	STATEMENT, 4(LAST_STATEMENT)	
				MOVL	STATEMENT, LAST_STATEMENT	
				BRB	1\$	: 0980
				RET		: 0995

; Routine Size: 83 bytes, Routine Base: \$CODE\$ + 0000



```

265 0996 1 1 ++
266 0997 1 1 Description: This parsing routine recognizes a 'statement', which is any
267 0998 1 1 of the major types of definitions that appear in a CLD.
268 0999 1 1
269 1000 1 1 Parameters: parent By reference, parent node of this construct.
270 1001 1 1
271 1002 1 1 Returns: The top-level node representing the statement.
272 1003 1 1
273 1004 1 1 Notes:
274 1005 1 1 --
275 1006 1 1
276 1007 1 1 GLOBAL ROUTINE cdu$statement(parent: ref node)
277 1008 2 2 = BEGIN
278 1009 2 2
279 1010 2 2 Local
280 1011 2 2 statement: ref node;
281 1012 2 2
282 1013 2 2
283 1014 2 2 ! Determine which kind of statement we have.
284 1015 2 2
285 1016 2 2 if token_is(tkn_k_symbol,'IDENT') then (
286 1017 2 2
287 1018 2 2 ! The IDENT statement supplies a string to be stashed in the object
288 1019 2 2 ! file. It conflicts with any existing statement.
289 1020 2 2
290 1021 2 2 if cdu$check_for_children(.parent,node_k_ident) then
291 1022 2 2 cdu$report_syntax_error(msg(cdu$_dupident));
292 1023 2 2
293 1024 2 2 ! The next token is the string. Save it in a node.
294 1025 2 2
295 1026 2 2 cdu$get_next_token(tkn_k_h_string);
296 1027 2 2 statement = cdu$create_node(node_k_ident,.cdu$gq_token[len],.cdu$gq_token[ptr]);
297 1028 2 2 cdu$token_must_be(tkn_k_string);
298 1029 2 2 return .statement;
299 1030 2 2 );
300 1031 2 2
301 1032 2 2 if token_is(tkn_k_symbol,'MODULE') then (
302 1033 2 2
303 1034 2 2 ! The MODULE statement supplies a symbol to be used as the name of the
304 1035 2 2 ! CLD object module. It conflicts with any existing statement.
305 1036 2 2
306 1037 2 2 if cdu$check_for_children(.parent,node_k_module) then
307 1038 2 2 cdu$report_syntax_error(msg(cdu$_dupmodule));
308 1039 2 2
309 1040 2 2 ! The next token is the symbol. Save it in a node.
310 1041 2 2
311 1042 2 2 cdu$get_next_token();
312 1043 2 2 statement = cdu$create_node(node_k_module,.cdu$gq_token[len],.cdu$gq_token[ptr]);
313 1044 2 2 cdu$token_must_be(tkn_k_symbol);
314 1045 2 2 return .statement;
315 1046 2 2 );
316 1047 2 2
317 1048 2 2 ! We must have a DEFINE statement. Get the next token and see what kind
318 1049 2 2 ! of a DEFINE it is. The appropriate syntax routine is called and returns
319 1050 2 2 ! the top-level node representing the statement.
320 1051 2 2
321 1052 2 2 cdu$token_must_be(tkn_k_symbol,ctext('DEFINE'));

```



```

322 1053 2 if token_is(tkn_k_symbol,'VERB') then
323 1054 2 statement = cdu$define_verb(.parent)
324 1055 2 else if token_is(tkn_k_symbol,'SYNTAX') then
325 1056 2 statement = cdu$define_syntax(.parent)
326 1057 2 else if token_is(tkn_k_symbol,'TYPE') then
327 1058 2 statement = cdu$define_type(.parent)
328 1059 2 else (
329 1060 2 cdu$report_syntax_error(msg(cdu$_invdefine));
330 1061 2 return cdu$create_node(node_k_error);
331 1062 2 );
332 1063 2 ! If there is already a definition with the same name, then we have a conflict.
333 1064 2 ! Tell the user about it.
334 1065 2
335 1066 2 if cdu$lookup_child(.parent,.statement[node_w_type],
336 1067 2 .statement[node_b_text_length],statement[node_t_text]) neq 0 then
337 1068 2 cdu$report_syntax_error(msg(cdu$_dupdef),1,statement[node_b_text_length]);
338 1069 2 return .statement;
339 1070 2
340 1071 2
341 1072 1 END;

```

```

.PSECT $PLITS$,NOWRT,NOEXE,2
      54 4E 45 44 49 00000 P.AAA: .ASCII \IDENT\
45 45 4C 55 44 4F 4D 00005 P.AAB: .ASCII \MODULE\
      4E 49 46 45 44 06 0000B P.AAC: .ASCII <6>\DEFINE\
      42 52 45 56 00012 P.AAD: .ASCII \VERB\
      58 41 54 4E 59 53 00016 P.AAE: .ASCII \SYNTAX\
      45 50 59 54 0001C P.AAF: .ASCII \TYPE\

.EXTRN CDU$_DUPIDENT, CDU$_DUPMODULE
.EXTRN CDU$_INVDEFINE, CDU$_DUPDEF

.PSECT $CODE$,NOWRT,2
      OFFC 00000
      5B 00000000G 00 9E 00002 MOVAB CDU$STATEMENT, Save R2,R3,R4,R5,R6,R7,R8,-
      5A 00000000G 00 9E 00009 MOVAB R9,R10,R11
      59 00000000G 00 9E 00010 MOVAB CDU$CHECK_FOR_CHILDREN, R11
      58 0000' CF 9E 00017 MOVAB CDU$GL_TOKEN_CLASS, R10
      57 00000000G 00 9E 0001C MOVAB CDU$CREATE_NODE, R9
      56 00000000G 00 9E 00023 MOVAB P.AAA, R8
      0D 6A D1 0002A MOVAB CDU$REPORT_SYNTAX_ERROR, R7
      50 66 D0 0002F MOVAB CDU$GQ_TOKEN+4, R6
      60 FC A6 2D 00032 CMPL CDU$GL_TOKEN_CLASS, #13
      68 00038 BNEQ 2$
      50 66 D0 0002F MOVL CDU$GQ_TOKEN+4, R0
      60 FC A6 2D 00032 CMPC5 CDU$GQ_TOKEN, (R0), #0, #5, P.AAA
      68 00038 BNEQ 2$
      02 DD 0003B PUSHL #2
      04 AC DD 0003D PUSHL PARENT
      6B 02 FB 00040 CALLS #2, CDU$CHECK_FOR_CHILDREN
      09 50 E9 00043 BLBC R0, 1$
      00000000G 8F DD 00046 PUSHL #CDU$_DUPIDENT
      67 01 FB 0004C CALLS #1, CDU$REPORT_SYNTAX_ERROR

```



				0C	DD	0004F	1\$:	PUSHL	#12		1026
				01	FB	00051		CALLS	#1, CDUSGET_NEXT_TOKEN		
				66	DD	00058		PUSHL	CDUSGQ_TOKEN+4		1027
			7E	FC	A6	3C	0005A	MOVZWL	CDUSGQ_TOKEN, -(SP)		
				02	DD	0005E		PUSHL	#2		
			69		03	FB	00060	CALLS	#3, CDUSCREATE_NODE		
			54		50	D0	00063	MOVL	R0, STATEMENT		
					0B	DD	00066	PUSHL	#11		1028
					3D	11	00068	BRB	4\$		
			0D		6A	D1	0006A	2\$:	CMPL	CDUSGL_TOKEN_CLASS, #13	1032
					42	12	0006D	BNEQ	5\$		
			50		66	D0	0006F	MOVL	CDUSGQ_TOKEN+4, R0		
06	00		60	FC	A6	2D	00072	CMPC5	CDUSGQ_TOKEN, (R0), #0, #6, P.AAB		
				05	A8		00078				
					35	12	0007A	BNEQ	5\$		
					03	DD	0007C	PUSHL	#3		1037
				04	AC	DD	0007E	PUSHL	PARENT		
			6B		02	FB	00081	CALLS	#2, CDUSCHECK_FOR_CHILDREN		
			09		50	E9	00084	BLBC	R0, 3\$		
					8F	DD	00087	PUSHL	#CDUS_DUPMODULE		1038
			67		01	FB	0008D	CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
			00		00	FB	00090	3\$:	CALLS	#0, CDUSGET_NEXT_TOKEN	1042
					66	DD	00097	PUSHL	CDUSGQ_TOKEN+4		1043
			7E	FC	A6	3C	00099	MOVZWL	CDUSGQ_TOKEN, -(SP)		
					03	DD	0009D	PUSHL	#3		
			69		03	FB	0009F	CALLS	#3, CDUSCREATE_NODE		
			54		50	D0	000A2	MOVL	R0, STATEMENT		
					0D	DD	000A5	PUSHL	#13		1044
			00		01	FB	000A7	4\$:	CALLS	#1, CDUSTOKEN_MUST_BE	
					0098	31	000AE	BRW	11\$		1045
				0B	A8	9F	000B1	5\$:	PUSHAB	P.AAC	1052
					0D	DD	000B4	PUSHL	#13		
			00		02	FB	000B6	CALLS	#2, CDUSTOKEN_MUST_BE		
					55	D4	000BD	CLRL	R5		1053
			0D		6A	D1	000BF	CMPL	CDUSGL_TOKEN_CLASS, #13		
					19	12	000C2	BNEQ	6\$		
					55	D6	000C4	INCL	R5		
			50		66	D0	000C6	MOVL	CDUSGQ_TOKEN+4, R0		
04	00		60	FC	A6	2D	000C9	CMPC5	CDUSGQ_TOKEN, (R0), #0, #4, P.AAD		
				12	A8		000CF				
					0A	12	000D1	BNEQ	6\$		
				04	AC	DD	000D3	PUSHL	PARENT		1054
			0000V	CF	01	FB	000D6	CALLS	#1, CDUSDEFINE_VERB		
					32	11	000DB	BRB	8\$		
			34		55	E9	000DD	6\$:	BLBC	R5, 9\$	1055
			50		66	D0	000E0	MOVL	CDUSGQ_TOKEN+4, R0		
06	00		60	FC	A6	2D	000E3	CMPC5	CDUSGQ_TOKEN, (R0), #0, #6, P.AAE		
				16	A8		000E9				
					0A	12	000EB	BNEQ	7\$		
				04	AC	DD	000ED	PUSHL	PARENT		1056
			0000V	CF	01	FB	000F0	CALLS	#1, CDUSDEFINE_SYNTAX		
					18	11	000F5	BRB	8\$		
			1A		55	E9	000F7	7\$:	BLBC	R5, 9\$	1057
			50		66	D0	000FA	MOVL	CDUSGQ_TOKEN+4, R0		
04	00		60	FC	A6	2D	000FD	CMPC5	CDUSGQ_TOKEN, (R0), #0, #4, P.AAF		
				1C	A8		00103				
					0D	12	00105	BNEQ	9\$		

PARSE1  
V04-000

E 8  
15-Sep-1984 23:46:44  
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[CDU.SRC]PARSE1.B32;1

Page 12  
(7)

0000V	CF	04	AC	DD	00107	PUSHL	PARENT	:	1058
	54		01	FB	0010A	CALLS	#1, CDUS\$DEFINE_TYPE	:	
			50	D0	0010F	8\$:	RO, STATEMENT	:	
			0F	11	00112	MOVL	10\$	:	
		00000000G	8F	DD	00114	9\$:	PUSHL	#CDUS\$ INVDEFINE	1060
	67		01	FB	0011A	CALLS	#1, CDUS\$REPORT_SYNTAX_ERROR	:	
			7E	D4	0011D	CLRL	-(SP)	:	1061
	69		01	FB	0011F	CALLS	#1, CDUS\$CREATE_NODE	:	
			04	00122	RET			:	
		11	A4	9F	00123	10\$:	PUSHAB	17(STATEMENT)	1068
	7E	10	A4	9A	00126	MOVZBL	16(STATEMENT), -(SP)	:	
	7E		64	3C	0012A	MOVZWL	(STATEMENT), -(SP)	:	
		04	AC	DD	0012D	PUSHL	PARENT	:	
00000000G	00		04	FB	00130	CALLS	#4, CDUS\$LOOKUP_CHILD	:	
			50	D5	00137	TSTL	RO	:	
			0E	13	00139	BEQL	11\$	:	
		10	A4	9F	0013B	PUSHAB	16(STATEMENT)	:	1069
			01	DD	0013E	PUSHL	#1	:	
		00000000G	8F	DD	00140	PUSHL	#CDUS\$ DUPDEF	:	
	67		03	FB	00146	CALLS	#3, CDUS\$REPORT_SYNTAX_ERROR	:	
	50		54	D0	00149	11\$:	MOVL	STATEMENT, RO	1070
			04	0014C	RET			:	1072

; Routine Size: 333 bytes, Routine Base: \$CODE\$ + 0053



```

: 343 1073 1 !++
: 344 1074 1 ! Description: This parsing routine recognizes the "define-verb" construct,
: 345 1075 1 ! which is used to define a new DCL verb.
: 346 1076 1 !
: 347 1077 1 ! Parameters: parent By reference, parent node of this construct.
: 348 1078 1 !
: 349 1079 1 ! Returns: By reference, the top-level node of the statement.
: 350 1080 1 !
: 351 1081 1 ! Notes:
: 352 1082 1 ! --
: 353 1083 1 !
: 354 1084 1 GLOBAL ROUTINE cdu$define_verb(parent: ref node)
: 355 1085 2 = BEGIN
: 356 1086 2
: 357 1087 2 local
: 358 1088 2 statement: ref node,
: 359 1089 2 clause: ref node,
: 360 1090 2 last_clause: ref node;
: 361 1091 2
: 362 1092 2
: 363 1093 2 ! The next token must be the name of the verb. Create a node to represent the
: 364 1094 2 ! statement and put the name in it.
: 365 1095 2
: 366 1096 2 cdu$get_next_token();
: 367 1097 2 statement = cdu$create_node(node_k_define_verb,.cdu$gq_token[len],.cdu$gq_token[ptr]);
: 368 1098 2 cdu$token_must_be(tkn_k_symbol);
: 369 1099 2
: 370 1100 2 ! Now we have a sequence of clauses which describe the verb.
: 371 1101 2
: 372 1102 2 loop (
: 373 1103 2 ! The clauses may be separated by commas.
: 374 1104 2
: 375 1105 2 skip_optional_token(tkn_k_comma);
: 376 1106 2
: 377 1107 2 ! Parse a clause. If there weren't any more, then we are done.
: 378 1108 2 ! Otherwise link the clause node on to the end of the clause chain.
: 379 1109 2
: 380 1110 2 clause = cdu$v_s_clause(.statement);
: 381 1111 2 if .clause eql 0 then exitloop;
: 382 1112 2 link_parent_to_child(statement,clause,last_clause);
: 383 1113 2 );
: 384 1114 2
: 385 1115 2 return .statement;
: 386 1116 2
: 387 1117 1 END;

```

INFO#250 L1:1112  
Referenced LOCAL symbol LAST\_CLAUSE is probably not initialized

55	00000000G	00	9E	00002	.ENTRY	CDU\$DEFINE VERB, Save R2,R3,R4,R5	: 1084
65	00000000G	00	FB	00009	MOVAB	CDU\$GET NEXT TOKEN, R5	: 1096
	00000000G	00	DD	0000C	CALLS	#0, CDU\$GET NEXT_TOKEN	: 1097
7E	00000000G	00	3C	00012	PUSHL	CDU\$GQ_TOKEN+4	:
					MOVZWL	CDU\$GQ_TOKEN, -(SP)	:

PARSE1  
V04-000

G 8  
15-Sep-1984 23:46:44  
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[CDU.SRC]PARSE1.B32;1 Page 14  
(8)

00000000G	00	04	DD	00019	PUSHL	#4			
	54	03	FB	0001B	CALLS	#3, CDU\$CREATE_NODE			
		50	D0	00022	MOVL	R0, STATEMENT			
00000000G	00	0D	DD	00025	PUSHL	#13			1098
	05	01	FB	00027	CALLS	#1, CDUSTOKEN_MUST_BE			
		00	D1	0002E	CMPL	CDU\$GL_TOKEN_CLASS, #5			1105
	65	03	12	00035	BNEQ	2\$			
		00	FB	00037	CALLS	#0, CDU\$GET_NEXT_TOKEN			
0000V	CF	54	DD	0003A	PUSHL	STATEMENT			1110
	53	01	FB	0003C	CALLS	#1, CDU\$V_S_CLAUSE			
		50	D0	00041	MOVL	R0, CLAUSE			
		14	13	00044	BEQL	5\$			1111
		08	A4	D5	TSTL	8(STATEMENT)			1112
		06	12	00049	BNEQ	3\$			
08	A4	53	D0	0004B	MOVL	CLAUSE, 8(STATEMENT)			
		04	11	0004F	BRB	4\$			
04	A2	53	D0	00051	MOVL	CLAUSE, 4(LAST_CLAUSE)			
	52	53	D0	00055	MOVL	CLAUSE, LAST_CLAUSE			
		D4	11	00058	BRB	1\$			1098
	50	54	D0	0005A	MOVL	STATEMENT, R0			1115
		04	0005D	RET					1117

; Routine Size: 94 bytes, Routine Base: \$CODE\$ + 01A0



```

389 1118 1  ++
390 1119 1  Description: This parsing routine recognizes the 'define-syntax' construct,
391 1120 1  which is used to alter the syntax of a command based on the
392 1121 1  presence of some qualifier.
393 1122 1
394 1123 1  Parameters: parent By reference, parent node of this constuct.
395 1124 1
396 1125 1  Returns: By reference, the top-level node of the statement.
397 1126 1
398 1127 1  Notes:
399 1128 1  --
400 1129 1
401 1130 1  GLOBAL ROUTINE cdu$define_syntax(parent: ref node)
402 1131 2  = BEGIN
403 1132 2
404 1133 2  local
405 1134 2  statement: ref node,
406 1135 2  clause: ref node,
407 1136 2  last_clause: ref node;
408 1137 2
409 1138 2
410 1139 2  ! The next token must be the name of the syntax definition. Create a node to
411 1140 2  ! represent the statement and put the name in it.
412 1141 2
413 1142 2  cdu$get_next_token();
414 1143 2  statement = cdu$create_node(node_k_define_syntax,.cdu$gq_token[len],.cdu$gq_token[ptr]);
415 1144 2  cdu$token_must_be(tkn_k_symbol);
416 1145 2
417 1146 2  ! Now we have a sequence of clauses which describe the new syntax.
418 1147 2
419 1148 2  loop (
420 1149 2  ! The clauses may be separated by commas.
421 1150 2
422 1151 2  skip_optional_token(tkn_k_comma);
423 1152 2
424 1153 2  ! Parse a clause. If there weren't any more, then we are done.
425 1154 2  ! Otherwise link the clause node on to the end of the clause chain.
426 1155 2
427 1156 2  clause = cdu$vs_clause(.statement);
428 1157 2  if .clause eql 0 then exitloop;
429 1158 2  link_parent_to_child(statement,clause,last_clause);
430 1159 2 );
431 1160 2
432 1161 2  return .statement;
433 1162 2
434 1163 1  END;

```

INFO#250 L1:1158  
Referenced LOCAL symbol LAST\_CLAUSE is probably not initialized

55	00000000G	00	003C 00000	.ENTRY	CDU\$DEFINE SYNTAX, Save R2,R3,R4,R5	: 1130
65	00000000G	00	9E 00002	MOVAB	CDU\$GET NEXT TOKEN, R5	: 1142
		00	FB 00009	CALLS	#0, CDU\$GET NEXT_TOKEN	: 1143
		00	DD 0000C	PUSHL	CDU\$GQ_TOKEN+4	

	7E	00000000G	00	3C	00012	MOVZWL	CDUSGQ_TOKEN, -(SP)	:	
			05	DD	00019	PUSHL	#5	:	
00000000G	00		03	FB	0001B	CALLS	#3, CDUSCREATE_NODE	:	
	54		50	D0	00022	MOVL	R0, STATEMENT	:	1144
			0D	DD	00025	PUSHL	#13	:	
00000000G	00		01	FB	00027	CALLS	#1, CDUSTOKEN_MUST_BE	:	
	05	00000000G	00	D1	0002E	1\$:	CDUSGL_TOKEN_CLASS, #5	:	1151
			03	12	00035	BNEQ	2\$	:	
	65		00	FB	00037	CALLS	#0, CDUSGET_NEXT_TOKEN	:	
			54	DD	0003A	2\$:	STATEMENT	:	1156
0000V	CF		01	FB	0003C	CALLS	#1, CDUSV_S_CLAUSE	:	
	53		50	D0	00041	MOVL	R0, CLAUSE	:	
			14	13	00044	BEQL	5\$	:	1157
		08	A4	D5	00046	TSTL	8(STATEMENT)	:	1158
			06	12	00049	BNEQ	3\$	:	
08	A4		53	D0	0004B	MOVL	CLAUSE, 8(STATEMENT)	:	
			04	11	0004F	BRB	4\$	:	
04	A2		53	D0	00051	3\$:	CLAUSE, 4(LAST_CLAUSE)	:	
	52		53	D0	00055	4\$:	CLAUSE, LAST_CLAUSE	:	
			D4	11	00058	BRB	1\$	:	1144
	50		54	D0	0005A	5\$:	STATEMENT, R0	:	1161
			04	0005D		RET		:	1163

; Routine Size: 94 bytes, Routine Base: \$CODE\$ + 01FE



```

436      1164 1 !++
437      1165 1 !Description: This parsing routine recognizes the 'define-type' construct,
438      1166 1 !which is used to define a set of keywords which some
439      1167 1 !qualifier can take as values.
440      1168 1
441      1169 1 !Parameters: parent By reference, parent node of this construct.
442      1170 1
443      1171 1 !Returns: By reference, the top-level node of the statement.
444      1172 1
445      1173 1 !Notes:
446      1174 1 !--
447      1175 1
448      1176 1 GLOBAL ROUTINE cdu$define_type(parent: ref node)
449      1177 2 = BEGIN
450      1178 2
451      1179 2 local
452      1180 2     statement: ref node,
453      1181 2     clause: ref node,
454      1182 2     last_clause: ref node;
455      1183 2
456      1184 2
457      1185 2 ! The next token must be the name of the type definition. Create a node to
458      1186 2 ! represent the statement and put the name in it.
459      1187 2
460      1188 2 cdu$get_next_token();
461      1189 2 statement = cdu$create_node(node_k_define_type,.cdu$gq_token[len],.cdu$gq_token[ptr]);
462      1190 2 cdu$token_must_be(tkn_k_symbol);
463      1191 2
464      1192 2 ! Now we have a sequence of clauses which describe the type keywords.
465      1193 2
466      1194 2 loop (
467      1195 2     ! The clauses may be separated by commas.
468      1196 2
469      1197 2     skip_optional_token(tkn_k_comma);
470      1198 2
471      1199 2     ! Parse a clause. If there weren't any more, then we are done.
472      1200 2     ! Otherwise link the clause node on to the end of the clause chain.
473      1201 2
474      1202 2     clause = cdu$type_clause(.statement);
475      1203 2     if .clause.eql 0 then exitloop;
476      1204 2     link_parent_to_child(statement,clause,last_clause);
477      1205 2 );
478      1206 2
479      1207 2 ! A type definition must include at least one keyword.
480      1208 2
481      1209 2 if not cdu$check_for_children(.statement,node_k_keyword) then
482      1210 2     cdu$report_syntax_error(msg(cdu$reqkey),1,statement[node_b_text_length]);
483      1211 2 return .statement;
484      1212 2
485      1213 1 END;

```

INFO#250 L1:1204  
Referenced LOCAL symbol LAST\_CLAUSE is probably not initialized

.EXTRN CDUS\_REQKEY

		003C 00000	.ENTRY	CDU\$DEFINE TYPE, Save R2,R3,R4,R5	: 1176
55	00000000G	00 9E 00002	MOVAB	CDU\$GET_NEXT_TOKEN, R5	:
65		00 FB 00009	CALLS	#0, CDU\$GET_NEXT_TOKEN	: 1188
	00000000G	00 DD 0000C	PUSHL	CDU\$GQ_TOKEN+4	: 1189
7E	00000000G	00 3C 00012	MOVZWL	CDU\$GQ_TOKEN, -(SP)	:
		06 DD 00019	PUSHL	#6	:
00000000G	00	03 FB 0001B	CALLS	#3, CDU\$CREATE_NODE	:
52		50 D0 00022	MOVL	R0, STATEMENT	:
		0D DD 00025	PUSHL	#13	: 1190
00000000G	00	01 FB 00027	CALLS	#1, CDU\$TOKEN_MUST_BE	:
05	00000000G	00 D1 0002E 1\$:	CMPL	CDU\$GL_TOKEN_CLASS, #5	: 1197
		03 12 00035	BNEQ	2\$	:
65		00 FB 00037	CALLS	#0, CDU\$GET_NEXT_TOKEN	:
		52 DD 0003A 2\$:	PUSHL	STATEMENT	: 1202
00000000G	00	01 FB 0003C	CALLS	#1, CDU\$TYPE_CLAUSE	:
54		50 D0 00043	MOVL	R0, CLAUSE	:
		14 13 00046	BEQL	5\$	: 1203
	08	A2 D5 00048	TSTL	8(STATEMENT)	: 1204
		06 12 0004B	BNEQ	3\$	:
08	A2	54 D0 0004D	MOVL	CLAUSE, 8(STATEMENT)	:
		04 11 00051	BRB	4\$	:
04	A3	54 D0 00053 3\$:	MOVL	CLAUSE, 4(LAST_CLAUSE)	:
53		54 D0 00057 4\$:	MOVL	CLAUSE, LAST_CLAUSE	:
		D2 11 0005A	BRB	1\$	: 1190
		18 DD 0005C 5\$:	PUSHL	#24	: 1209
		52 DD 0005E	PUSHL	STATEMENT	:
00000000G	00	02 FB 00060	CALLS	#2, CDU\$CHECK_FOR_CHILDREN	:
12		50 E8 00067	BLBS	R0, 6\$	:
	10	A2 9F 0006A	PUSHAB	16(STATEMENT)	: 1210
		01 DD 0006D	PUSHL	#1	:
	00000000G	8F DD 0006F	PUSHL	#CDU\$ REQKEY	:
00000000G	00	03 FB 00075	CALLS	#3, CDU\$REPORT_SYNTAX_ERROR	:
50		52 D0 0007C 6\$:	MOVL	STATEMENT, R0	: 1211
		04 0007F	RET		: 1213

; Routine Size: 128 bytes, Routine Base: \$CODE\$ + 025C



```

487 1214 1 !++
488 1215 1 ! Description: This syntax routine recognizes the 'v-s-clause' construct,
489 1216 1 ! which are the clauses used to describe a verb or syntax
490 1217 1 ! definition.
491 1218 1
492 1219 1 ! Parameters: parent By reference, parent node of this constuct.
493 1220 1
494 1221 1 ! Returns: The top-level node representing the clause, or zero if
495 1222 1 ! there is no clause we recognize.
496 1223 1
497 1224 1 ! Notes:
498 1225 1 ! --
499 1226 1
500 1227 1 GLOBAL ROUTINE cdu$v_s_clause(parent: ref node)
501 1228 2 = BEGIN
502 1229 2
503 1230 2 local
504 1231 2     clause: ref node,
505 1232 2     item: ref node,
506 1233 2     last_item: ref node;
507 1234 2
508 1235 2
509 1236 2 ! Determine which kind of clause we have.
510 1237 2
511 1238 2 if token_is(tkn_k_symbol,'CLIFLAGS') then (
512 1239 2
513 1240 2     ! We have a CLIFLAGS clause. Create a parent node for the flags.
514 1241 2
515 1242 2     clause = cdu$create_node(node_k_cliflags);
516 1243 2     cdu$get_next_token();
517 1244 2
518 1245 2     ! We have a parenthesized list of CLI flags.
519 1246 2     ! Eat the open parenthesis. Then sit in a loop which recognizes at
520 1247 2     ! least one item, along with any others separated by commas. Finally,
521 1248 2     ! eat the close parenthesis. All of the items are chained as children
522 1249 2     ! of the clause.
523 1250 2
524 1251 2     cdu$token_must_be(tkn_k_open_paren);
525 1252 2     loop (
526 1253 2         item = cdu$cli_flag(.clause);
527 1254 2         link_parent_to_child(clause,item,last_item);
528 1255 2         if not token_is(tkn_k_comma) then exitloop;
529 1256 2         cdu$get_next_token();
530 1257 2     );
531 1258 2     cdu$token_must_be(tkn_k_close_paren);
532 1259 2     return .clause;
533 1260 2 );
534 1261 2
535 1262 2 if token_is(tkn_k_symbol,'CLIROUTINE') then (
536 1263 2
537 1264 2     ! We have a CLIROUTINE clause. It conflicts with any existing
538 1265 2     ! CLIROUTINE, IMAGE, or ROUTINE clause.
539 1266 2
540 1267 2     if cdu$check_for_children(.parent,node_k_cliroutine,node_k_image,node_k_routine) then
541 1268 2         cdu$report_syntax_error(msg(cdu$_confouting));
542 1269 2
543 1270 2     ! Next we have a symbol which is the name of an internal CLI

```

```

544 1271 3      ! routine that processes the command. Create a node with the
545 1272      ! symbol.
546 1273
547 1274      cdu$get_next_token();
548 1275      clause = cdu$create_node(node_k_cliroutine,.cdu$gq_token[ptr],.cdu$gq_token[ptr]);
549 1276      cdu$token_must_be(tkn_k_symbol);
550 1277      return .clause;
551 1278 );
552 1279
553 1280 if token_is(tkn_k_symbol,'DISALLOW') then (
554 1281
555 1282      ! We have a DISALLOW clause. It conflicts with any existing
556 1283      ! NODISALLOWS clause.
557 1284
558 1285      if cdu$check_for_children(.parent,node_k_nodisallows) then
559 1286          cdu$report_syntax_error(msg(cdu$_confnodis));
560 1287
561 1288      ! Create a node for the clause.
562 1289
563 1290      clause = cdu$create_node(node_k_disallow);
564 1291      cdu$get_next_token();
565 1292
566 1293      ! We now have a boolean expression. Chain it on to the parent node.
567 1294
568 1295      clause[node_l_child] = cdu$bool_expr();
569 1296      return .clause;
570 1297 );
571 1298
572 1299 if token_is(tkn_k_symbol,'NODISALLOWS') then (
573 1300
574 1301      ! We have a NODISALLOWS clause. It is represented by a node.
575 1302      ! This clause conflicts with any existing DISALLOW clause.
576 1303
577 1304      if cdu$check_for_children(.parent,node_k_disallow) then
578 1305          cdu$report_syntax_error(msg(cdu$_confdis));
579 1306
580 1307      cdu$get_next_token();
581 1308      return cdu$create_node(node_k_nodisallows);
582 1309 );
583 1310
584 1311 if token_is(tkn_k_symbol,'IMAGE') then (
585 1312
586 1313      ! We have an IMAGE clause. It conflicts with any existing
587 1314      ! CLIROUTINE, IMAGE, or ROUTINE clause.
588 1315
589 1316      if cdu$check_for_children(.parent,node_k_cliroutine,node_k_image,node_k_routine) then
590 1317          cdu$report_syntax_error(msg(cdu$_confouting));
591 1318
592 1319      ! Now we have a string or an h-string which is the spec of the image
593 1320      ! to be activated for this command.
594 1321
595 1322      cdu$get_next_token(tkn_k_h_string);
596 1323
597 1324      ! Strip trailing blanks and tabs
598 1325
599 1326      while ch$rchar(.cdu$gq_token[ptr]+.cdu$gq_token[ptr]-1) EQL %C' '
600 1327          or ch$rchar(.cdu$gq_token[ptr]+.cdu$gq_token[ptr]-1) EQL %C'

```



```

601      1328      3      do cdu$gq_token[len] = .cdu$gq_token[len] - 1;
602      1329      clause = cdu$create_node(node_k_image,.cdu$gq_token[len],.cdu$gq_token[ptr]);
603      1330
604      1331      ! Make sure the string isn't too long.
605      1332
606      1333      if .clause[node_b_text_length] gtru cmd_k_max_image-1 then
607      1334          cdu$report_syntax_error(msg(cdu$_image_len),1,cmd_k_max_image-1);
608      1335
609      1336      cdu$token_must_be(tkn_k_string);
610      1337      return .clause;
611      1338  );
612      1339
613      1340  if token_is(tkn_k_symbol,'OUTPUTS') then (
614      1341
615      1342      ! We have an OUTPUTS clause. It conflicts with an existing clause.
616      1343
617      1344      if cdu$check_for_children(.parent,node_k_outputs) then
618      1345          cdu$report_syntax_error(msg(cdu$_confoutputs));
619      1346
620      1347      ! Create a node to represent the clause.
621      1348
622      1349      clause = cdu$create_node(node_k_outputs);
623      1350      cdu$get_next_token();
624      1351
625      1352      ! We have a parenthesized list of output items.
626      1353      ! Eat the open parenthesis. Then sit in a loop which recognizes at
627      1354      ! least one item, along with any others separated by commas. Finally,
628      1355      ! eat the close parenthesis. All of the items are chained as children
629      1356      ! of the clause.
630      1357
631      1358      cdu$token_must_be(tkn_k_open_paren);
632      1359      loop (
633      1360          item = cdu$create_node(node_k_outputs_item,.cdu$gq_token[len],.cdu$gq_token[ptr]);
634      1361          cdu$token_must_be(tkn_k_symbol);
635      1362          link_parent_to_child(clause,item,last_item);
636      1363          if not token_is(tkn_k_comma) then exitloop;
637      1364          cdu$get_next_token();
638      1365      );
639      1366      cdu$token_must_be(tkn_k_close_paren);
640      1367      return .clause;
641      1368  );
642      1369
643      1370  if token_is(tkn_k_symbol,'PARAMETER') then (
644      1371
645      1372      ! We have a PARAMETER definition. It conflicts with any existing
646      1373      ! NOPARAMETERS clause.
647      1374
648      1375      if cdu$check_for_children(.parent,node_k_noparameters) then
649      1376          cdu$report_syntax_error(msg(cdu$_confnoparm));
650      1377
651      1378      ! The first thing is the parameter name. Create a node for it.
652      1379
653      1380      cdu$get_next_token();
654      1381      clause = cdu$create_node(node_k_parameter,.cdu$gq_token[len],.cdu$gq_token[ptr]);
655      1382
656      1383      ! Ensure that the parameter name is in the form Pn.
657      1384

```



```

: 658      1385 4      (bind
: 659      1386 4          name = .cdu$gq_token[ptr]: vector[,byte];
: 660      1387 4
: 661      1388 4      if .cdu$gq_token[len] nequ 2 or
: 662      1389 4          .name[0] nequ 'P' or
: 663      1390 4          .name[1] lssu '1' or .name[1] gtru '0'+cmd_k_max_parms then
: 664      1391 4          cdu$report_syntax_error(msg(cdu$_invparm));
: 665      1392 4      );
: 666      1393 4      cdu$token_must_be(tkn_k_symbol);
: 667      1394 4
: 668      1395 4      ! We have a list of parameter definition clauses. Each is optionally
: 669      1396 4      ! preceded by a comma. All of the items are chained as children of the
: 670      1397 4      ! main parameter clause.
: 671      1398 4
: 672      1399 4      loop (
: 673      1400 4          skip_optional_token(tkn_k_comma);
: 674      1401 4          item = cdu$param_clause(.clause);
: 675      1402 4          if .item eql 0 then exitloop;
: 676      1403 4          link_parent_to_child(clause,item,last_item);
: 677      1404 4      );
: 678      1405 4
: 679      1406 4      return .clause;
: 680      1407 4 );
: 681      1408 4
: 682      1409 4 if token_is(tkn_k_symbol,'NOPARAMETERS') then (
: 683      1410 4
: 684      1411 4     ! We have a NOPARAMETERS clause. It is represented by a node.
: 685      1412 4     ! This clause conflicts with any existing PARAMETER clause.
: 686      1413 4
: 687      1414 4     if cdu$check_for_children(.parent,node_k_parameter) then
: 688      1415 4         cdu$report_syntax_error(msg(cdu$_confparm));
: 689      1416 4
: 690      1417 4     cdu$get_next_token();
: 691      1418 4     return cdu$create_node(node_k_noparameters);
: 692      1419 4 );
: 693      1420 4
: 694      1421 4 if token_is(tkn_k_symbol,'PREFIX') then (
: 695      1422 4
: 696      1423 4     ! We have an PREFIX clause. It conflicts with any existing clause.
: 697      1424 4
: 698      1425 4     if cdu$check_for_children(.parent,node_k_prefix) then
: 699      1426 4         cdu$report_syntax_error(msg(cdu$_duppprefix));
: 700      1427 4
: 701      1428 4     ! Now we have a symbol which is the prefix. Create a node and put
: 702      1429 4     ! the symbol in it.
: 703      1430 4
: 704      1431 4     cdu$get_next_token();
: 705      1432 4     clause = cdu$create_node(node_k_prefix,.cdu$gq_token[len],.cdu$gq_token[ptr]);
: 706      1433 4     cdu$token_must_be(tkn_k_symbol);
: 707      1434 4     return .clause;
: 708      1435 4 );
: 709      1436 4
: 710      1437 4 if token_is(tkn_k_symbol,'QUALIFIER') then (
: 711      1438 4
: 712      1439 4     ! We have a QUALIFIER definition. It conflicts with any existing
: 713      1440 4     ! NOQUALIFIERS clause.
: 714      1441 4

```



```

: 715      1442      3      if cdu$check_for_children(.parent,node_k_noqualifiers) then
: 716      1443      3      cdu$report_syntax_error(msg(cdu$_confnoqual));
: 717      1444      3
: 718      1445      3      ! The next item is the name of the qualifier. Create a node for it.
: 719      1446      3
: 720      1447      3      cdu$get_next_token();
: 721      1448      3      clause = cdu$create_node(node_k_qualifier,.cdu$gq_token[len],.cdu$gq_token[ptr]);
: 722      1449      3
: 723      1450      3      ! The definition also conflicts with any existing definition of the
: 724      1451      3      ! same name. However, we can't check for this in V4 because of
: 725      1452      3      ! layered products with the ZZZZ qualifier placeholder hacks in
: 726      1453      3      ! their CLDs (VMS' fault, not theirs).
: 727      1454      3
: 728      1455      3      if cdu$lookup_child(.parent,node_k_qualifier,
: 729      1456      3      .clause[node_b_text_length],clause[node_t_text]) neq 0 then
: 730      1457      3      cdu$report_syntax_error(msg(cdu$_dupqual),1,clause[node_b_text_length]);
: 731      1458      3      cdu$token_must_be(tkn_k_symbol);
: 732      1459      3
: 733      1460      3      ! We have a list of qualifier definition clauses. Each is optionally
: 734      1461      3      ! preceded by a comma. All of the items are chained as children of the
: 735      1462      3      ! main qualifier clause.
: 736      1463      3
: 737      1464      4      loop (
: 738      1465      4      skip_optional_token(tkn_k_comma);
: 739      1466      4      item = cdu$qual_clause(.clause);
: 740      1467      4      if .item eq 0 then exitloop;
: 741      1468      4      link_parent_to_child(clause,item,last_item);
: 742      1469      4      );
: 743      1470      3
: 744      1471      3      return .clause;
: 745      1472      2      );
: 746      1473      2
: 747      1474      3      if token_is(tkn_k_symbol,'NOQUALIFIERS') then (
: 748      1475      3
: 749      1476      3      ! We have a NOQUALIFIERS clause. It is represented by a node.
: 750      1477      3      ! This clause conflicts with any existing QUALIFIER clause.
: 751      1478      3
: 752      1479      3      if cdu$check_for_children(.parent,node_k_qualifier) then
: 753      1480      3      cdu$report_syntax_error(msg(cdu$_confqual));
: 754      1481      3
: 755      1482      3      cdu$get_next_token();
: 756      1483      3      return cdu$create_node(node_k_noqualifiers);
: 757      1484      3      );
: 758      1485      3
: 759      1486      3      if token_is(tkn_k_symbol,'ROUTINE') then (
: 760      1487      3
: 761      1488      3      ! We have an ROUTINE clause. It conflicts with any existing
: 762      1489      3      ! CLIROUTINE, IMAGE, or ROUTINE clause.
: 763      1490      3
: 764      1491      3      if cdu$check_for_children(.parent,node_k_cliroutine,node_k_image,node_k_routine) then
: 765      1492      3      cdu$report_syntax_error(msg(cdu$_confrouting));
: 766      1493      3
: 767      1494      3      ! Next we have a symbol which is the name of the user routine which
: 768      1495      3      ! can process this command. Create a node containing the symbol.
: 769      1496      3
: 770      1497      3      cdu$get_next_token();
: 771      1498      3      clause = cdu$create_node(node_k_routine,.cdu$gq_token[len],.cdu$gq_token[ptr]);

```

```

: 772      1499      3      cdu$token_must_be(tkn_k_symbol);
: 773      1500      3      return .c[ause];
: 774      1501      2      );
: 775      1502      2
: 776      1503      2      if token_is(tkn_k_symbol,'SYNONYM') then (
: 777      1504      2
: 778      1505      2          ! We have a SYNONYM clause. It specifies a symbol which is a synonym
: 779      1506      2          ! for the verb name. Create a node containing the symbol.
: 780      1507      2
: 781      1508      2          cdu$get_next_token();
: 782      1509      2          clause = cdu$create_node(node_k_synonym,.cdu$gq_token[len],.cdu$gq_token[ptr]);
: 783      1510      2          cdu$token_must_be(tkn_k_symbol);
: 784      1511      2          return .c[ause];
: 785      1512      2      );
: 786      1513      2
: 787      1514      2      ! We don't have a clause that we understand, so there probably aren't any more.
: 788      1515      2
: 789      1516      2      return 0;
: 790      1517      2
: 791      1518      1      END;

```

INFO#250 L1:1254  
Referenced LOCAL symbol LAST\_ITEM is probably not initialized

```

.PSECT $SPLITS,NOWRT,NOEXE,2
      45  4E  53  47  41  4C  46  49  4C  43  00020 P.AAG: .ASCII \CLIFLAGS\
      53  57  4F  4C  4C  41  53  49  44  4F  4E  00028 P.AAH: .ASCII \CLIRoutine\
      53  57  4F  4C  4C  41  53  49  44  4F  4E  00032 P.AAI: .ASCII \DISALLOW\
      53  57  4F  4C  4C  41  53  49  44  4F  4E  0003A P.AAJ: .ASCII \NODISALLOWS\
      53  57  4F  4C  4C  41  53  49  44  4F  4E  00045 P.AAK: .ASCII \IMAGE\
      53  57  4F  4C  4C  41  53  49  44  4F  4E  0004A P.AAL: .ASCII \OUTPUTS\
      53  52  45  52  45  54  45  4D  41  52  41  50  00051 P.AAM: .ASCII \PARAMETER\
      53  52  45  52  45  54  45  4D  41  52  41  50  0005A P.AAN: .ASCII \NOPARAMETERS\
      53  52  45  52  45  58  49  46  45  52  50  00066 P.AAO: .ASCII \PREFIX\
      53  52  45  52  45  49  46  49  4C  41  55  51  0006C P.AAP: .ASCII \QUALIFIER\
      53  52  45  49  46  49  4C  41  55  51  4F  4E  00075 P.AAQ: .ASCII \NOQUALIFIERS\
      53  52  45  49  46  45  4E  49  54  55  4F  52  00081 P.AAR: .ASCII \ROUTINE\
      53  52  45  49  46  4D  59  4E  4F  4E  59  53  00088 P.AAS: .ASCII \SYNONYM\

.EXTRN CDUS_CONFROUTING
.EXTRN CDUS_CONFNODIS, CDUS_CONFDIS
.EXTRN CDUS_IMAGELEN, CDUS_CONFOUTPUTS
.EXTRN CDUS_CONFNOPARM
.EXTRN CDUS_INVPARM, CDUS_CONFPAARM
.EXTRN CDUS_DUPPREFIX, CDUS_CONFNOQUAL
.EXTRN CDUS_CONFQUAL

.PSECT $CODE$,NOWRT,2
      OFFC 00000
      5B 00000000G 00 9E 00002
      5A 00000000G 00 9E 00009
      59 00000000G 00 9E 00010
      58 00000000G 00 9E 00017

.ENTRY CDUSV S CLAUSE, Save R2,R3,R4,R5,R6,R7,R8,- 1227
      R9,R10,R11
      MOVAB CDUSCHECK FOR CHILDREN, R11
      MOVAB CDUSREPORT SYNTAX ERROR, R10
      MOVAB CDUSGL TOKEN CLASS, R9
      MOVAB CDUSGET_NEXT_TOKEN, R8

```



		57	00000000G	00	9E	0001E	MOVAB	CDUSGQ_TOKEN, R7		
		0D		69	D1	00025	CMPL	CDUSGL_TOKEN_CLASS, #13	1238	
				51	12	00028	BNEQ	5\$		
08	00	50	04	A7	D0	0002A	MOVL	CDUSGQ_TOKEN+4, R0		
		60		67	2D	0002E	CMPC5	CDUSGQ_TOKEN, (R0), #0, #8, P.AAG		
			0000'	CF		00033				
				43	12	00036	BNEQ	5\$		
				07	DD	00038	PUSHL	#7	1242	
		00000000G	00	01	FB	0003A	CALLS	#1, CDUSCREATE_NODE		
			54	50	D0	00041	MOVL	R0, CLAUSE		
			68	00	FB	00044	CALLS	#0, CDUSGET_NEXT_TOKEN	1243	
				07	DD	00047	PUSHL	#7	1251	
		00000000G	00	01	FB	00049	CALLS	#1, CDUSTOKEN_MUST_BE		
			54	DD	00050	1\$:	PUSHL	CLAUSE	1253	
		00000000G	00	01	FB	00052	CALLS	#1, CDUSCLI_FLAG		
			56	50	D0	00059	MOVL	R0, ITEM		
			08	A4	D5	0005C	TSTL	8(CLAUSE)	1254	
				06	12	0005F	BNEQ	2\$		
	08	A4		56	D0	00061	MOVL	ITEM, 8(CLAUSE)		
				04	11	00065	BRB	3\$		
	04	A5		56	D0	00067	2\$:	MOVL	ITEM, 4(LAST_ITEM)	
		55		56	D0	0006B	3\$:	MOVL	ITEM, LAST_ITEM	
		05		69	D1	0006E	CMPL	CDUSGL_TOKEN_CLASS, #5	1255	
				03	13	00071	BEQL	4\$		
				0197	31	00073	BRW	22\$		
		68		00	FB	00076	4\$:	CALLS	#0, CDUSGET_NEXT_TOKEN	1256
				D5	11	00079	BRB	1\$	1251	
		0D		69	D1	0007B	5\$:	CMPL	CDUSGL_TOKEN_CLASS, #13	1262
				34	12	0007E	BNEQ	7\$		
		50	04	A7	D0	00080	MOVL	CDUSGQ_TOKEN+4, R0		
0A	00	60		67	2D	00084	CMPC5	CDUSGQ_TOKEN, (R0), #0, #10, P.AAH		
			0000'	CF		00089				
				26	12	0008C	BNEQ	7\$		
				12	DD	0008E	PUSHL	#18	1267	
				0A	DD	00090	PUSHL	#10		
				08	DD	00092	PUSHL	#8		
			04	AC	DD	00094	PUSHL	PARENT		
		6B		04	FB	00097	CALLS	#4, CDUSCHECK_FOR_CHILDREN		
		09		50	E9	0009A	BLBC	R0, 6\$		
			00000000G	8F	DD	0009D	PUSHL	#CDUS_CONFRUOTING	1268	
		6A		01	FB	000A3	CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
		68		00	FB	000A6	6\$:	CALLS	#0, CDUSGET_NEXT_TOKEN	1274
			04	A7	DD	000A9	PUSHL	CDUSGQ_TOKEN+4	1275	
		7E		67	3C	000AC	MOVZWL	CDUSGQ_TOKEN, -(SP)		
				08	DD	000AF	PUSHL	#8		
				0357	31	000B1	BRW	52\$		
		0D		69	D1	000B4	7\$:	CMPL	CDUSGL_TOKEN_CLASS, #13	1280
				3F	12	000B7	BNEQ	9\$		
		50	04	A7	D0	000B9	MOVL	CDUSGQ_TOKEN+4, R0		
08	00	60		67	2D	000BD	CMPC5	CDUSGQ_TOKEN, (R0), #0, #8, P.AAI		
			0000'	CF		000C2				
				31	12	000C5	BNEQ	9\$		
				2A	DD	000C7	PUSHL	#42	1285	
			04	AC	DD	000C9	PUSHL	PARENT		
		6B		02	FB	000CC	CALLS	#2, CDUSCHECK_FOR_CHILDREN		
		09		50	E9	000CF	BLBC	R0, 8\$		
			00000000G	8F	DD	000D2	PUSHL	#CDUS_CONFNODIS	1286	

		6A		01	FB	000D8		CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
				09	DD	000DB	8\$:	PUSHL	#9		1290
	00000000G	00		01	FB	000DD		CALLS	#1, CDUSCREATE_NODE		
		54		50	DO	000E4		MOVL	RO, CLAUSE		
		68		00	FB	000E7		CALLS	#0, CDUSGET_NEXT_TOKEN		1291
	00000000G	00		00	FB	000EA		CALLS	#0, CDUSBOOC_EXPR		1295
		08	A4	50	DO	000F1		MOVL	RO, 8(CLAUSE)		
				0326	31	000F5		BRW	54\$		1296
		0D		69	D1	000F8	9\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1299
				2A	12	000FB		BNEQ	11\$		
0B		50	04	A7	DO	000FD		MOVL	CDUSGQ_TOKEN+4, RO		
	00	60		67	2D	00101		CMPC5	CDUSGQ_TOKEN, (RO), #0, #11, P.AAJ		
			0000'	CF		00106					
				1C	12	00109		BNEQ	11\$		
				09	DD	0010B		PUSHL	#9		1304
			04	AC	DD	0010D		PUSHL	PARENT		
6B				02	FB	00110		CALLS	#2, CDUSCHECK_FOR_CHILDREN		
09				50	E9	00113		BLBC	RO, 10\$		
	00000000G			8F	DD	00116		PUSHL	#CDUS_CONFDIS		1305
6A				01	FB	0011C		CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
68				00	FB	0011F	10\$:	CALLS	#0, CDUSGET_NEXT_TOKEN		1307
				2A	DD	00122		PUSHL	#42		1308
				0286	31	00124		BRW	48\$		
0D				69	D1	00127	11\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1311
				6B	12	0012A		BNEQ	17\$		
50		04		A7	DO	0012C		MOVL	CDUSGQ_TOKEN+4, RO		
05	00	50		67	2D	00130		CMPC5	CDUSGQ_TOKEN, (RO), #0, #5, P.AAK		
			0000'	CF		00135					
				5D	12	00138		BNEQ	17\$		
				12	DD	0013A		PUSHL	#18		1316
				0A	DD	0013C		PUSHL	#10		
				08	DD	0013E		PUSHL	#8		
			04	AC	DD	00140		PUSHL	PARENT		
6B				04	FB	00143		CALLS	#4, CDUSCHECK_FOR_CHILDREN		
09				50	E9	00146		BLBC	RO, 12\$		
	00000000G			8F	DD	00149		PUSHL	#CDUS_CONFRUIMG		1317
6A				01	FB	0014F		CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
				0C	DD	00152	12\$:	PUSHL	#12		1322
68				01	FB	00154		CALLS	#1, CDUSGET_NEXT_TOKEN		
50				67	3C	00157	13\$:	MOVZWL	CDUSGQ_TOKEN, RO		1326
50		04		A7	CO	0015A		ADDL2	CDUSGQ_TOKEN+4, RO		
20		FF		A0	91	0015E		CMPB	-1(RO), #32		
				06	13	00162		BEQL	14\$		
09		FF		A0	91	00164		CMPB	-1(RO), #9		1327
				04	12	00168		BNEQ	15\$		
				67	B7	0016A	14\$:	DECW	CDUSGQ_TOKEN		1328
				E9	11	0016C		BRB	13\$		
		04		A7	DD	0016E	15\$:	PUSHL	CDUSGQ_TOKEN+4		1329
7E				67	3C	00171		MOVZWL	CDUSGQ_TOKEN, -(SP)		
				0A	DD	00174		PUSHL	#10		
00000000G	00			03	FB	00176		CALLS	#3, CDUSCREATE_NODE		
	54			50	DO	0017D		MOVL	RO, CLAUSE		
3F		10		A4	91	00180		CMPB	16(CLAUSE), #63		1333
				0D	1B	00184		BLEQU	16\$		
				3F	DD	00186		PUSHL	#63		1334
				01	DD	00188		PUSHL	#1		
			00000000G	8F	DD	0018A		PUSHL	#CDUS_IMAGELEN		



07	00	6A	03	FB	00190	CALLS	#3, CDUS\$REPORT_SYNTAX_ERROR	1336	
			0B	DD	00193	PUSHL	#11	1336	
			78	11	00195	BRB	23\$	1340	
		0D	69	D1	00197	CMPL	CDUS\$GL_TOKEN_CLASS, #13	1340	
			76	12	0019A	BNEQ	24\$		
		50	A7	D0	0019C	MOVL	CDUS\$GQ_TOKEN+4, R0		
		60	67	2D	001A0	CMPC5	CDUS\$GQ_TOKEN, (R0), #0, #7, P.AAL		
			CF		001A5				
			68	12	001A8	BNEQ	24\$		
			0B	DD	001AA	PUSHL	#11	1344	
			04	AC	DD	001AC	PUSHL	PARENT	
		6B	02	FB	001AF	CALLS	#2, CDUS\$CHECK_FOR_CHILDREN		
		09	50	E9	001B2	BLBC	R0, 18\$		
			8F	DD	001B5	PUSHL	#CDUS\$ CONFOUTPUTS	1345	
		6A	01	FB	001BB	CALLS	#1, CDUS\$REPORT_SYNTAX_ERROR	1349	
			0B	DD	001BE	PUSHL	#11		
		00000000G	00	01	FB	001C0	CALLS	#1, CDUS\$CREATE_NODE	
			54	50	D0	001C7	MOVL	R0, CLAUSE	
		68	00	FB	001CA	CALLS	#0, CDUS\$GET_NEXT_TOKEN	1350	
			07	DD	001CD	PUSHL	#7	1358	
		00000000G	00	01	FB	001CF	CALLS	#1, CDUS\$TOKEN_MUST_BE	
			A7	DD	001D6	PUSHL	CDUS\$GQ_TOKEN+4	1360	
		7E	67	3C	001D9	MOVZWL	CDUS\$GQ_TOKEN, -(SP)		
			0C	DD	001DC	PUSHL	#12		
		00000000G	00	03	FB	001DE	CALLS	#3, CDUS\$CREATE_NODE	
			50	D0	001E5	MOVL	R0, ITEM		
		00000000G	00	0D	DD	001E8	PUSHL	#13	1361
			01	FB	001EA	CALLS	#1, CDUS\$TOKEN_MUST_BE		
			A4	D5	001F1	TSTL	8(CLAUSE)	1362	
			06	12	001F4	BNEQ	20\$		
		08	A4	56	D0	001F6	MOVL	ITEM, 8(CLAUSE)	
			04	11	001FA	BRB	21\$		
		04	A5	56	D0	001FC	MOVL	ITEM, 4(LAST_ITEM)	
			55	56	D0	00200	MOVL	ITEM, LAST_ITEM	
			05	69	D1	00203	CMPL	CDUS\$GL_TOKEN_CLASS, #5	1363
				05	12	00206	BNEQ	22\$	
		68	00	FB	00208	CALLS	#0, CDUS\$GET_NEXT_TOKEN	1364	
			C9	11	0020B	BRB	19\$	1358	
			08	DD	0020D	PUSHL	#8	1366	
			0205	31	0020F	BRW	53\$		
		0D	69	D1	00212	CMPL	CDUS\$GL_TOKEN_CLASS, #13	1370	
			0C	12	00215	BNEQ	25\$		
		50	A7	D0	00217	MOVL	CDUS\$GQ_TOKEN+4, R0		
		60	67	2D	0021B	CMPC5	CDUS\$GQ_TOKEN, (R0), #0, #9, P.AAM		
			CF		00220				
			03	13	00223	BEQL	26\$		
			0083	31	00225	BRW	35\$		
			0E	DD	00228	PUSHL	#14	1375	
			04	AC	DD	0022A	PUSHL	PARENT	
		6B	02	FB	0022D	CALLS	#2, CDUS\$CHECK_FOR_CHILDREN		
		09	50	E9	00230	BLBC	R0, 27\$		
			8F	DD	00233	PUSHL	#CDUS\$ CONFNOPARM	1376	
		6A	01	FB	00239	CALLS	#1, CDUS\$REPORT SYNTAX ERROR		
		68	00	FB	0023C	CALLS	#0, CDUS\$GET NEXT_TOKEN	1380	
			A7	DD	0023F	PUSHL	CDUS\$GQ_TOKEN+4	1381	
		7E	67	3C	00242	MOVZWL	CDUS\$GQ_TOKEN, -(SP)		
			0D	DD	00245	PUSHL	#13		

			00000000G	00	03	FB	00247	CALLS	#3, CDUSCREATE_NODE		
				54	50	DO	0024E	MOVL	RO, CLAUSE		
				50	04	A7	DO	00251	MOVL	CDUSGQ_TOKEN+4, RO	1386
				02	67	B1	00255	CMPW	CDUSGQ_TOKEN, #2		1388
					12	12	00258	BNEQ	28\$		
			50	8F	60	91	0025A	CMPB	(RO), #80		1389
					0C	12	0025E	BNEQ	28\$		
				31	01	A0	91	00260	CMPB	1(RO), #49	1390
					06	1F	00264	BLSSU	28\$		
				38	01	A0	91	00266	CMPB	1(RO), #56	
					09	1B	0026A	BLEQU	29\$		
			00000000G		8F	DD	0026C	PUSHL	#CDUS INVPARM		1391
				6A	01	FB	00272	CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
					0D	DD	00275	PUSHL	#13		1393
			00000000G	00	01	FB	00277	CALLS	#1, CDUSTOKEN_MUST_BE		
				05	69	D1	0027E	CMPL	CDUSGL_TOKEN_CLASS, #5		1400
					03	12	00281	BNEQ	31\$		
				68	00	FB	00283	CALLS	#0, CDUSGET_NEXT_TOKEN		
					54	DD	00286	PUSHL	CLAUSE		1401
			00000000G	00	01	FB	00288	CALLS	#1, CDUSPARAM_CLAUSE		
				56	50	DO	0028F	MOVL	RO, ITEM		
					03	12	00292	BNEQ	32\$		1402
					0187	31	00294	BRW	54\$		
					08	A4	D5	00297	TSTL	8(CLAUSE)	1403
					06	12	0029A	BNEQ	33\$		
			08	A4	56	DO	0029C	MOVL	ITEM, 8(CLAUSE)		
					04	11	002A0	BRB	34\$		
			04	A5	56	DO	002A2	MOVL	ITEM, 4(LAST_ITEM)		
				55	56	DO	002A6	MOVL	ITEM, LAST_ITEM		
					D3	11	002A9	BRB	30\$		1393
				0D	69	D1	002AB	CMPL	CDUSGL_TOKEN_CLASS, #13		1409
					2A	12	002AE	BNEQ	37\$		
				50	04	A7	DO	002B0	MOVL	CDUSGQ_TOKEN+4, RO	
0C		00		60	67	2D	002B4	CMPC5	CDUSGQ_TOKEN, (RO), #0, #12, P.AAN		
					0000	CF	002B9				
					1C	12	002BC	BNEQ	37\$		
					0D	DD	002BE	PUSHL	#13		1414
					04	AC	DD	002C0	PUSHL	PARENT	
			68		02	FB	002C3	CALLS	#2, CDUSCHECK_FOR_CHILDREN		
			09		50	E9	002C6	BLBC	RO, 36\$		
			00000000G		8F	DD	002C9	PUSHL	#CDUS CONFPARM		1415
				6A	01	FB	002CF	CALLS	#1, CDUSREPORT SYNTAX ERROR		
				68	00	FB	002D2	CALLS	#0, CDUSGET_NEXT_TOKEN		1417
					0E	DD	002D5	PUSHL	#14		1418
					00D3	31	002D7	BRW	48\$		
				0D	69	D1	002DA	CMPL	CDUSGL_TOKEN_CLASS, #13		1421
					30	12	002DD	BNEQ	39\$		
				50	04	A7	DO	002DF	MOVL	CDUSGQ_TOKEN+4, RO	
06		00		60	67	2D	002E3	CMPC5	CDUSGQ_TOKEN, (RO), #0, #6, P.AAO		
					0000	CF	002E8				
					22	12	002EB	BNEQ	39\$		
					0F	DD	002ED	PUSHL	#15		1425
					04	AC	DD	002EF	PUSHL	PARENT	
			68		02	FB	002F2	CALLS	#2, CDUSCHECK_FOR_CHILDREN		
			09		50	E9	002F5	BLBC	RO, 38\$		
			00000000G		8F	DD	002F8	PUSHL	#CDUS DUPPREFIX		1426
				6A	01	FB	002FE	CALLS	#1, CDUSREPORT_SYNTAX_ERROR		



		68		00	FB	00301	38\$:	CALLS	#0, CDUSGET_NEXT_TOKEN		1431
		7E	04	A7	DD	00304		PUSHL	CDUSGQ_TOKEN+4		1432
				67	3C	00307		MOVZWL	CDUSGQ_TOKEN, -(SP)		
				0F	DD	0030A		PUSHL	#15		
				00FC	31	0030C		BRW	52\$		
		0D		69	D1	0030F	39\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1437
				6D	12	00312		BNEQ	46\$		
09	00	50	04	A7	D0	00314		MOVL	CDUSGQ_TOKEN+4, R0		
		60		67	2D	00318		CMPC5	CDUSGQ_TOKEN, (R0), #0, #9, P.AAP		
				0000'	CF	0031D					
				5F	12	00320		BNEQ	46\$		
				11	DD	00322		PUSHL	#17		1442
			04	AC	DD	00324		PUSHL	PARENT		
		6B		02	FB	00327		CALLS	#2, CDUSCHECK_FOR_CHILDREN		
		09		50	E9	0032A		BLBC	R0, 40\$		
				00000000G	8F	DD	0032D	PUSHL	#CDUS_CONFNOQUAL		1443
		6A		01	FB	00333		CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
		68		00	FB	00336	40\$:	CALLS	#0, CDUSGET_NEXT_TOKEN		1447
			04	A7	DD	00339		PUSHL	CDUSGQ_TOKEN+4		1448
		7E		67	3C	0033C		MOVZWL	CDUSGQ_TOKEN, -(SP)		
				10	DD	0033F		PUSHL	#16		
		00000000G	00	03	FB	00341		CALLS	#3, CDUSCREATE_NODE		
			54	50	D0	00348		MOVL	R0, CLAUSE		
				0D	DD	0034B		PUSHL	#13		1458
		00000000G	00	01	FB	0034D		CALLS	#1, CDUSTOKEN_MUST_BE		
			05	69	D1	00354	41\$:	CMPL	CDUSGL_TOKEN_CLASS, #5		1465
				03	12	00357		BNEQ	42\$		
		68		00	FB	00359		CALLS	#0, CDUSGET_NEXT_TOKEN		
				54	DD	0035C	42\$:	PUSHL	CLAUSE		1466
		00000000G	00	01	FB	0035E		CALLS	#1, CDUSQUAL_CLAUSE		
			56	50	D0	00365		MOVL	R0, ITEM		
				03	12	00368		BNEQ	43\$		1467
				00B1	31	0036A		BRW	54\$		
			08	A4	D5	0036D	43\$:	TSTL	8(CLAUSE)		1468
				06	12	00370		BNEQ	44\$		
		08	A4	56	D0	00372		MOVL	ITEM, 8(CLAUSE)		
				04	11	00376		BRB	45\$		
		04	A5	56	D0	00378	44\$:	MOVL	ITEM, 4(LAST_ITEM)		
			55	56	D0	0037C	45\$:	MOVL	ITEM, LAST_ITEM		
				D3	11	0037F		BRB	41\$		1458
		0D		69	D1	00381	46\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1474
				2F	12	00384		BNEQ	49\$		
		50	04	A7	D0	00386		MOVL	CDUSGQ_TOKEN+4, R0		
0C	00	60		67	2D	0038A		CMPC5	CDUSGQ_TOKEN, (R0), #0, #12, P.AAQ		
				0000'	CF	0038F					
				21	12	00392		BNEQ	49\$		
				10	DD	00394		PUSHL	#16		1479
			04	AC	DD	00396		PUSHL	PARENT		
		6B		02	FB	00399		CALLS	#2, CDUSCHECK_FOR_CHILDREN		
		09		50	E9	0039C		BLBC	R0, 47\$		
				00000000G	8F	DD	0039F	PUSHL	#CDUS_CONFQUAL		1480
		6A		01	FB	003A5		CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
		68		00	FB	003A8	47\$:	CALLS	#0, CDUSGET_NEXT_TOKEN		1482
				11	DD	003AB		PUSHL	#17		1483
		00000000G	00	01	FB	003AD	48\$:	CALLS	#1, CDUSCREATE_NODE		
				04	003B4			RET			
		0D		69	D1	003B5	49\$:	CMPL	CDUSGL_TOKEN_CLASS, #13		1486

07	00	50	04	33	12	003B8	BNEQ	51\$		
		60		A7	D0	003BA	MOVL	CDUSGQ_TOKEN+4, R0		
			0000'	67	2D	003BE	CMPC5	CDUSGQ_TOKEN, (R0), #0, #7, P.AAR		
				CF		003C3				
				25	12	003C6	BNEQ	51\$		
				12	DD	003C8	PUSHL	#18		1491
				0A	DD	003CA	PUSHL	#10		
				08	DD	003CC	PUSHL	#8		
			04	AC	DD	003CE	PUSHL	PARENT		
		6B		04	FB	003D1	CALLS	#4, CDUSCHECK_FOR_CHILDREN		
		09		50	E9	003D4	BLBC	R0, 50\$		
			00000000G	8F	DD	003D7	PUSHL	#CDUS_CONFROUTING		1492
		6A		01	FB	003DD	CALLS	#1, CDUSREPORT_SYNTAX_ERROR		
		68		00	FB	003E0	CALLS	#0, CDUSGET_NEXT_TOKEN		1497
			04	A7	DD	003E3	PUSHL	CDUSGQ_TOKEN+4		1498
		7E		67	3C	003E6	MOVZWL	CDUSGQ_TOKEN, -(SP)		
				12	DD	003E9	PUSHL	#18		
				1E	11	003EB	BRB	52\$		
		0D		69	D1	003ED	CMPL	CDUSGL_TOKEN_CLASS, #13		1503
				30	12	003F0	BNEQ	55\$		
		50	04	A7	D0	003F2	MOVL	CDUSGQ_TOKEN+4, R0		
		60		67	2D	003F6	CMPC5	CDUSGQ_TOKEN, (R0), #0, #7, P.AAS		
			0000'	CF		003FB				
				22	12	003FE	BNEQ	55\$		
		68		00	FB	00400	CALLS	#0, CDUSGET_NEXT_TOKEN		1508
			04	A7	DD	00403	PUSHL	CDUSGQ_TOKEN+4		1509
		7E		67	3C	00406	MOVZWL	CDUSGQ_TOKEN, -(SP)		
				34	DD	00409	PUSHL	#52		
			00000000G	03	FB	0040B	CALLS	#3, CDUSCREATE_NODE		
		54		50	D0	00412	MOVL	R0, CLAUSE		
				0D	DD	00415	PUSHL	#13		1510
			00000000G	01	FB	00417	CALLS	#1, CDUSTOKEN_MUST_BE		
		50		54	D0	0041E	MOVL	CLAUSE, R0		1511
					04	00421	RET			
				50	D4	00422	CLRL	R0		1516
					04	00424	RET			1518

: Routine Size: 1061 bytes, Routine Base: \$CODE\$ + 02DC

: 792 1519 1  
: 793 1520 1 END  
: 794 1521 0 ELUDOM

# PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1793	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	143	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)



PARSE1  
V04-000

K 9  
15-Sep-1984 23:46:44  
14-Sep-1984 11:58:26

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[CDU.SRC]PARSE1.B32;1 Page 31  
(11)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
;\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	4	0	1000	00:01.9

: Information: 5  
: Warnings: 0  
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:PARSE1/OBJ=OBJ\$:PARSE1 MSRC\$:PARSE1/UPDATE=(ENH\$:PARSE1)

: Size: 1793 code + 147 data bytes  
: Run Time: 00:34.5  
: Elapsed Time: 01:13.1  
: Lines/CPU Min: 2642  
: Lexemes/CPU-Min: 19323  
: Memory Used: 345 pages  
: Compilation Complete



0044 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

